R E P O R T R E S U M E S

ED 016 398 24 EM 005 971

OPERATING SOFTWARE FOR A COMPUTER-BASED INSTRUCTION SYSTEM.

BY- KAUPE, ARTHUR F., JR.
PITTSBURGH UNIV., PA., LEARNING RES. AND DEV. CTR.

REPORT NUMBER TR-3 PUB DATE 10 MAR 66
REPORT NUMBER BR-5-0253
REPORT NUMBER RR-66-5K00-TEACH-R1
EDRS PRICE MF-$0.25 HC-$0.72 16P.

DESCRIPTORS- *COMPUTER ASSISTED INSTRUCTION, *DIGITAL COMPUTERS, ADMINISTRATION, RESOURCE ALLOCATIONS, PROGRAM DEVELOPMENT, INDIVIDUALIZED PROGRAMS

FOUR SOFTWARE COMPONENTS (COMPILER, INTERPRETER, EXECUTIVE, SERVICE) OF A COMPUTER-BASED INSTRUCTION SYSTEM ARE DESCRIBED. (LH)

OPERATING SOFTWARE FOR A

COMPUTER-BASED INSTRUCTION SYSTEM

Arthur F. Kaupe

of the

Westinghouse Research Laboratories

in conjunction with the

Learning Research and Development Center

University of Pittsburgh

March, 1966

The research and development reported herein was performed in part pursuant to a contract with the United States Office of Education, Department of Health, Education, and Welfare under the provisions of the Cooperative Research Program.

OPERATING SOFTWARE FOR A COMPUTER-BASED INSTRUCTION SYSTEM

by

A. F. Kaupe, Jr.

Abstract

There is an obvious need to develop and program courses which are to be conducted by a Computer-Based Instruction System (CBIS). If such systems are to be economically feasible, there is a not so obvious need for the development and programming operating software for a CBIS. This operating software can be viewed as being composed of four systems - complier, interpreter, executive, and service - which are used primarily to translate course programs for machine execution, direct system response to individual students, allocate resources, and perform necessary administration. Those systems are described in general terms intended to provide the reader some understanding of the problems involved, and to indicate the interrelationships among these four systems.

Research Report 66-5K00-TEACH-R1
Proprietary Class 2

March 10, 1966

OPERATING SOFTWARE FOR A COMPUTER-BASED INSTRUCTION SYSTEM

by

A. F. Kaupe, Jr.

INTRODUCTION

The research, experimentation, and development in the areas of programmed instruction, teaching machines, and digital computers has led to the concept of Computer Aided Instruction (CAI) as a means to cope with the shortage of teachers, overabundance of students, and increased responsibilities which plague our educational system. In the school using CAI, each student would spend a portion (say 25% - 33%) of his school day receiving instruction through a student console of a Computer-Based Instruction System (CBIS), while the remainder of his day would be spent in more conventional school situations. Each CBIS includes, in addition to the digital computer which is the heart of the system, (1) a number (say 50 - 150) of student consoles which are used by the student to communicate with the CBIS, (2) mass storage devices of various characteristics in which course material and student records are preserved, and (3) central input/output equipment used primarily for administrative purposes.

A CBIS is, however, more than just the equipment or hardware; it requires considerable programming or software to become a reality. The most obvious programming problem is the preparation of courses in which the CBIS is to provide instruction. Despite the fact that most of the programming for a CBIS will necessarily be course-programming, it is in some ways the smallest programming problem. More accurately, there are a number of other programming problems - the operating software - which must be solved if there is to be a feasible solution to the problem of preparing courses. This report has been written to indicate the nature of these subsidiary programming problems and to outline the relationships between them.

To discuss the operating software, we have identified four major components -- the compiler, the interpreter, the executive, and the service system. While, for ease of description, we shall frequently use these labels as if each component were independent and had well defined boundaries, they are in fact very closely related and, among the last three in particular, have very fuzzy boundaries. Before proceeding, we give some indication of the primary functions each component is to perform.

As the reader may be aware, writing computer programs in machine or machine-oriented languages is an expensive and time consuming operation. In order to make programming costs reasonable, problem- or human-oriented programming languages have been developed for other areas and must be developed for preparing courses. The compiler is that portion of the software which translates the programs written in the special language to a form suitable for execution by the machine. This translation process is, normally, performed but once at a given CBIS installation and results in an interpretive program.

The interpretive program is the form in which a course is stored for use. The interpreter examines this program to determine the sequence of machine operations to be performed. This is also a translation process; however, it is performed as the student progresses, and only that portion of the course is further translated into machine operations as may be required. The use of the interpreter (which could be avoided) is desirable primarily to reduce the need for storage. The interpretive form of the course requires significantly less memory than a corresponding machine operation form. While it is quite true that space is required for the interpreter which must, in effect, contain a copy of the machine language equivalent for each operation in the interpretive program, the CBIS can be arranged so that but one copy of the interpreter is required even if 100 students working on 100 different courses are using the CBIS simultaneously.

The use of a single computer to work with a large number of students simultaneously requires a large amount of coordination. It is unreasonable to expect the person writing the course to cope with the

many details required to effect this coordination. There are three basic areas where coordination in the use of system resources is required: (1) control of input/output equipment; (2) allocation of available memory; and (3) allocation of computer processing capabilities -- which are the primary responsibility of the executive. The term executive is used because in a sense, the decisions made by this component are policy decisions and only minimal administrative functions are performed within the executive system.

The compiler and the interpreter will also be required to perform a small amount of administration; however, it is the service system which will perform most of the administrative tasks. The service system is both the glue that holds the system together and the oil that makes it run smoothly. A simple example of this dual role occurs when a student sits down at a console and makes his presence known. It is the executive which would first recognize the presence of a new student since he represents a demand for facilities under control of the executive. However, the executive would immediately call upon the service system to identify the student and his overall requirements. The service system having determined for example, that the student wished to continue work on an arithmetic course would locate, through its own master records, the information the interpreter will need to work with the student and have the executive system copy the necessary information in memory where the interpreter can use it. It would then advise the executive system of any special requirements where the executive system would, after fulfilling the requirements, permit the interpreter to proceed.

The Compiler

The preparation of courses to be administered by a CBIS will normally be accomplished by educational specialists. These specialists should be relatively unconcerned about many of the features of the CBIS. For example, the fact that 100 students may be using the system at the same time may be interesting but it is essentially immaterial insofar as the educational specialist is concerned. While programming a course, he should be able to work as if the CBIS had nothing to do but conduct the course he is writing. His primary concern should be the educational

effectiveness of his program and thus must be able to specify the types and timing of presentations to the student, to make decisions based on responses from the student, to prepare and manipulate both semi-permanent and temporary records concerning the student's behavior, and to communicate with teachers, proctors, and administrators especially when unusual situations develop.

In order that the course programmer can program with this freedom; it will be necessary to develop a special programming language -- and a computer program for translating the programs written in this special language. Preliminary efforts to develop a "standard" language have been sponsored by the Office of Naval Research and are being coordinated by the University of Pittsburgh's Learning Research and Development Center. While it is far too early to predict the form which this language will take, it is already quite clear that the translator for this language will require full exploitation of all techniques for translating programs now known and will require development of yet more sophisticated techniques since audio and video as well as the normal digital information will be an integral part of virtually every course program.

One approach, not necessarily the best, would be to first write the program, identifying in some manner audio and video messages. The output from the compiler might then be (1) the program used to conduct the course, (2) a program which directs and controls the recording of the necessary audio information in a form suitable for use by the first program, and (3) a program which similarly directs and controls the recording of video information. Except to note that it is desirable that these latter programs be executable on a CBIS, we shall not concern curselves with the latter programs. Insofar as the first program generated by the compiler is concerned, it must also contain many, if not most, of the uncertainties of the program written by the educational specialists. That is to say, while the "code" generated by the compiler may reflect in some ways the fact that the program is to be useable by many different students, using different stations at different times, neither a particular student nor a particular station has been specified.

The primary function of the compiler is not to resolve these undefined aspects of the course program, but rather to convert the program into a form suitable for efficient execution. The most important properties of the compiler and language are that they provide the programmer with basic operations which reflect the programmer's general requirements rather than machine structure and that they provide convenient mechanisms for defining new "higher level" operations which are appropriate in particular courses, or portions of courses. As an example where this latter property is needed, consider the idea of "evaluating" the student's response. This concept can arise since, at least as far as the computer per se is concerned, the response may be just a sequence of characters. When we write a number, there are infinitely many sequences of characters which represent the same number, e.g.; 1492, 1492.0, 1.492×10^3, etc. are all representations of the same number. In a course in mathematics, any one of these forms may be equally acceptable; however, in a history course, only the first form might be considered acceptable. Another way of viewing this issue is to say that the form of the response may affect the validity of the response. Where considerable latitude as to form of response is or must be allowed, there is a need to evaluate the response. Since different courses, or even different sections in the same course, may require special concepts of evaluation which are not of general value, it is clearly desirable that the compiler provide some mechanism whereby the educational programmer can define his special concept and then use the defined concept just as if it were part of the language.

One characteristic of natural languages (English, French, German, etc.) is that they continually evolve -- they are not static, fixed, well-defined languages. Most programming languages, however, have been relatively static; the changes in them have tended to be revolutionary rather than evolutionary. This has been due in large part to the techniques used to prepare the compilers for these languages in that little thought or provision for possible changes has been reflected in the compiler design so that incorporating changes in the language is generally a tedious, time-consuming, expensive, and sometimes risky operation. On the other hand, a programming language developed within the next two years is extremely

unlikely to be a satisfactory programming language after five years. This is not to say that one won't be able to use the language but rather that the results of research into educational strategies and experience in using the language will show deficiencies in the language which must be corrected if future course programming costs are not to become too great.

Yet another aspect of the compiler that must be considered is diagnostic facilities. One technique used in connection with a translator for a simple, self-contained student console was to simulate the program during the translation process. Such an approach is quite feasible with the Skinner or linear programs which the self-contained unit uses, but is not likely to be adequate by itself in dealing with programs having complex logical structures. Exactly what diagnostic features other than detection of syntax or grammatical errors can and should be incorporated into the compiler is not at all clear.

A related problem is that of correcting errors in a program once they have been detected. If a course program represents, as it probably should, 20 - 40 hours of student study, recompiling a complete course to correct a minor error affecting only some 10 or 15 minute portion of the course is almost certainly going to be uneconomical. Thus, the compiler must have provision for recompiling segments of a course as corrections to already existing courses. This technique, sometimes called differential or incremental compiling, should however by implemented in a "natural" way, i.e., it would be undesirable to have the course programmer insert artificial markers merely to facilitate differential compiling. Closely related to this problem of differential compiling is yet another segmentation problem which results from the fact that there will be in general insufficient directly accessible memory to contain the complete course at any one time. Fortunately, it is not necessary to provide this much memory; however, it will be necessary for the compiler to segment the program into units consistent with the amount of directly accessible memory which will be available -- typically 100 - 200 words per student using the system.

The fact that the program which is produced by the compiler must still contain many undefined items (which station, which student, etc.), that the program must be segmented both to ease the problems of differential compiling, and to cope with the small amount of available memory, all suggest that this program will be not a sequence of machine instructions but rather collections of tables (most likely in the form of linked lists) to which are assigned names. Some tables and some entries in other tables may not actually exist in the program produced by the compiler, e.g., a table in which is to be recorded the specific station and specific student with which the course is working. This, however, is immaterial so long as other parts of the software can and do prepare these tables at an appropriate time.

The Interpreter

Among the tables prepared by the compiler is a subcollection which in effect specif the sequence of operations to be performed, including initiation of presentations to the student, analysis of responses from the student, and specification of how various decisions are to be made. The processing of this subcollection, which might be called the control tables of the program, is the primary function of the interpreter.

In the course of processing the control tables for a program, the interpreter will frequently be required to obtain information from or insert information in the other tables making up the course. This is the major reason for assigning names to the various tables for so long as a current directory of the whereabouts of various tables is maintained, the interpreter can easily refer to any portion of the course provided only that it knows the name of the table containing the information.

In general, the interpreter does not itself perform most of the operations specified in the control table; this is done in the most part by the executive and the service system. The interpreter merely determines by scanning the control tables what is to be done and initiates the particular operation by calling upon the appropriate portion of the other systems. It is due to this that it is a conceptually straightforward process to use but one copy of the interpreter no matter how many students may be using the system.

If, when a particular student starts working on a particular course a master table is prepared in which is recorded the point to which processing of the course has progressed, and the names, directly or indirectly, of all pertinent tables, then in order for the interpreter to process two or more systems will require (conceptually) nothing more than having the interpreter work with different master tables. In practice, somewhat more than this will usually be required - the exact amount of administrative detail involved in switching the interpreter from one master table to another being very strongly dependent upon structure of the computer being used. Some of the larger computers, such as the Burroughs B-5500 have incorporated this concept of a master table in the design of the computer so that nearly all administration is handled automatically, however, this is not usually the case with the smaller computers such as might be used in a CBIS.

Since, by and large, the interpreter has essentially no direct contact with the world outside the computer, it is difficult to give a more accurate description of its functions and problems. This situation is complicated further by the fact that a precise description will depend strongly upon the computer, and developments in educational strategy. There is, for example, good reason for expecting that some functions initially performed by the compiler may later be more efficiently performed by the interpreter. This should not be too surprising in that the interpreter may, at least in part, be considered a translator just as is the compiler. Thus, when courses are developed with portions which are used quite infrequently, it may well be more efficient to perform only basic editing of the program by the compiler and defer the final translation process until the material is actually required. Thus, as we warned the reader, the functional boundary between the interpreter and the compiler is quite indefinite. Operationally, there is a major distinction between these systems in that the compiler is used only to prepare a course for computer operation and does not of itself have any part in the teaching process, while the interpreter processes the output of the compiler to control the teaching process and is actively in use so long as students are using the system.

The Executive

Most modern electronic digital computers have a limited ability to perform a number of independent operations at the same time. Usually, this ability consists of the possibility of having a moderate (2 - 8) number of information transfer operations occurring simultaneously with a small number (1 - 4) of independent computation operations. If 100 or so students are using the system at one time, it is obvious that not all students can be serviced concurrently. Fortunately, this is not necessary if the computer itself is sufficiently fast; however, it does mean that there will generally be a number of students waiting for various operations to be performed at any one instant of time.

It is to cope with these "queuing" problems that the CBIS must have an executive. Generally speaking, there are three resources which must somehow be allocated among the various students: 1) space in memory, 2) communications channels, and 3) processor capacity. More importantly, these allocations must generally be made dynamically since the demands for these resources will vary from course to course, student to student, and, even given a fixed course-student pairing, from time to time. In a sense, when stated in terms of allocating resources, the problem of designing the executive is straightforward in that considerable help is available from such disciplines as queuing theory. Unfortunately, the problem is not so straightforward in that the information required to perform a meaningful a priori analysis is not available. The issue is further complicated by the fact that some of the resources to be allocated will be required to perform the allocations. In particular, memory space and processor capacity will be required by the executive and one must be careful that to insure that the cost in resources used by a particular executive is not more than what might be gained by the more "efficient" use of resources -- a question which is frequently asked but seldom answered by more than a wave of the hands.

However, the most troublesome problem is that the value of a given executive must be measured in psychological values. There exist many systems today which exhibit this resource sharing feature and which

allow direct communication between the system and a human being through keyboards, etc. Many, if not most, of these systems have response times in the order of seconds; for example, a recent advertisement for a computer system indicated that with 30 consoles, the system would "respond in two to three seconds". For many purposes, this response time is more than adequate; however, in one course programmed on an embryo CBIS at the University of Pittsburgh's Learning Research and Development Laboratory, the system must frequently respond in less than .1 second. Fortunately, not all courses necessarily require this fast response, not is it even always required within a course; however, the executive must be capable of providing the required response. Unfortunately, determination of the required response is a psychological problem since the affect of slow response must be evaluated in terms of its effects on interest, motivation, and learning rate of the student.

To further complicate the issue, the executive is usually something more than just the programming labeled the executive - in particular, the design of the executive usually will determine certain conventions which must be observed by the other components of the operating software in order to insure correct timing of various operations. For example, when a student depresses a key on a console, this may result in an "interrupt" signal being given to the computer. Normally the occurrence of an interrupt will automatically activate the executive to take appropriate action; however, situations can arise where immediate recognition of this interrupt cannot be permitted until some other, usually related, processing has been completed by some other portion of the system. Thus that part of the system must be able to defer recognition, or at least processing of the interrupt. The determination of these situations is, in fact, closely related to the design of the executive and must be closely coordinated since they may have a great effect on the ability of the executive to allocate resources effectively.

The Service System

We have already indicated that the service system is the glue that holds the CBIS together and the oil that makes it run smoothly. One

could break the service system into a number of subsystems; however, for our present purposes we choose not to do so. If you get the impression that the service system is comprised of a great deal of miscellaneous functions, then just consider the service or support units of any complex organization where you will find a similar situation.

One major responsibility of the service system is the records administration. It is in a sense the filing clerk and librarian for the CBIS. The importance of this function is correlated to the use of names for the tables which represent a course. As long as references to information are by name, the interpreter does not need to know where the information is kept so long as someone - the service system - can and does make the information available as needed. The service system then is charged with maintaining the records of all kinds -- student records, course records, courses, etc. This in turn means that the records are readily accessible by programs other than teaching programs, which is another important justification for this central filing system. A teacher may, using one of the consoles, inspect or analyze individual student records; progress reports can be prepared by yet other programs; classroom schedules prepared by yet another program, etc.

Another important function of the service system is interfacing between the interpreter and the executive. There are interfacing aspects in the record keeping functions mentioned above; however, there are more direct interfacing problems, primarily associated with the form in which information must be transmitted. The "logical information units" with which the interpreter expect to work may differ from the "physical information units" with which the executive must work, requiring that some form of editing or other manipulation be performed when transmitting information between these systems. We assign this function, which could be given to either the interpreter or the executive, to the service system on the grounds that the interpreter is and should be organized with concepts dealing with courses and units of information appropriate thereto while the executive is and should be organized with concepts dealing with the actual hardware and corresponding units of information. If this is done,

then the two systems will not, in general, mate with one another -- hence the service system -- however, the programming of the two components (interpreter and executive) can more easily proceed in parallel. More important than programming convenience is that the interface functions performed by the service routine are appropriate objects for study to determine whether these functions might not be more economically performed by hardware in a later version of the system.

Another consideration is that this holding of the interpreter to course concepts should simplify the introduction of new concepts in teaching techniques. This last is just a reflection of the lesson learned in computer programming that frequently the longer one can be indefinite about the meaning of a term, the more flexible one can be. One example of this is the concept of evaluation described earlier. By not giving the concept a fixed meaning, and instead, providing tools so that the course programmer can give a local definition, then the more useful the concept is. There are programming techniques available whereby this definition process can be held off until one is actually faced with performing an "evaluation" giving even more flexibility. While in the final analysis it does not matter where the necessary transformations are performed, it is conceptually cleaner to isolate this operation.

Another major function of the service system has to do with various malfunctions. Many, if not most, CBIS malfunctions (whether they be hardware or software) will be detected by other parts of the system. Certain of these will be such that the only safe procedure is to shut the CBIS down immediately. Most cases will permit less drastic handling ranging from a simple retry of an operation which was not successfully completed to discontinuing use of a given unit of equipment, notifying the maintenance personnel of the trouble, and, when appropriate, telling the student to move to another station.

We have already noted that the service system acts as a receptionist-inquiry clerk. There are a number of other important administrative functions for which the service system is responsible; however, rather than attempt to itemize them, we can categorize them -- the service system

is responsible for all those administrative processes which are necessary adjoints to the teaching process but which are not themselves part of this process. Given an ultimate system specification, it is conceivable that a service system might be unnecessary; but so long as CBIS systems continue to evolve, the concept of a service system as an independent entity is extremely useful, both during the initial design and while modifying the system.

SUMMARY

We have given a brief overview of the four primary components of the operating software for a Computer Based Instruction System. Briefly, these are:

1. The compiling system which serves to reduce the cost both in dollars and elapsed time of preparing courses, by processing programs for execution by
2. The interpretive system which initiates the processes specified by the course programmer;
3. The executive system which allocates the resources available in the CBIS; and
4. The service system which performs most of the purely administrative functions required by the other systems and, at times, acts as an interface between other components.

A detailed report on the design of an executive is in preparation and will be available in the near future. Studies of the other systems are in progress and reports will be made available as appropriate.